

Why talk about rotations in a Games Class?

TT,

TI

- Didn't really get to do it in 559
- · Can't let you leave without knowing about Quaternions
- Curious rift between theory and practice
 - Highly mathematical piece adopted early in Games
 - Workable solutions despite (and maybe better than) theory
- Really are useful!
 - Camera control / navigation
 - Rigid body dynamics
 - Articulated figure animation / Skinning

What is a rotation



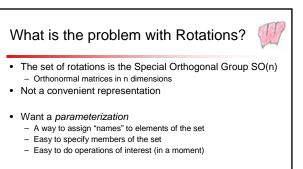
- A transformation Rⁿ -> Rⁿ, f(x) that a few properties
 - It has a "zero", such that f(0) = 0
 - It preserves "distances" such that |a-b| = |f(a)-f(b)|
 - It preserves "handedness"
- · From these properties, you can prove some others
 - It preserves (relative) angles
 - It is a LINEAR Transformation
 - It can be represented as an ortho-normal matrix
 - Rotations have unique inverses
 - The identity is a rotation
- · Rotations are important
 - Rigid motions
 - Viewpoint control

Important facts about rotations

- Closed under composition
- If A&B are rotations, AB is a rotation, as is BA
- Wrap around (not R)
- Non-Commutative AB != BA
- Associative (AB)C = A(BC)

A Detail

- Rotation is a transformation relative
- Orientation is an absolute configuration
- Rotation -> Orientation
 What happens when you apply a rotation to the identity
- Rotation between two orientations
 A⁻¹B
- · Can think of this as local coordinates of the first object



- Fundamental theorem of topology
 - Any representation in Rⁿ will have problems
 it has a different "shape"
 - Singularities, Redundancies, ...
 - Hairy Ball Theorem

What might you want to do with Rotations?



- Specify easily
- Represent compactly
- Make sure that you have a rotation (no errors)
- Find the inverse
- Transform points
- Interpolate 2 rotations
- · Blend n rotations
- Average n rotations
- Do other linear operations
 Filter, splines, ...
- No singularities (measure distances)

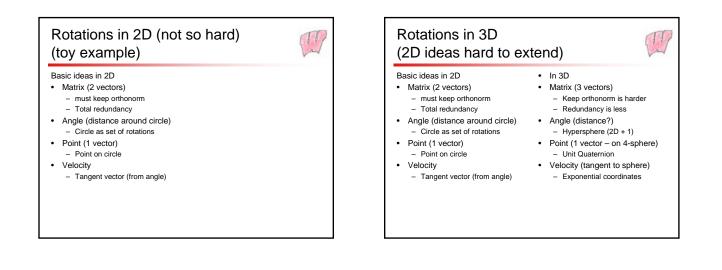
Matrix is a representation for rotations

- Any rotation can be stored as a matrix (1 -> 1)
- Not every matrix is a rotation
 Can ask about the "closest" rotation matrix to a given on
 Projection onto a subset

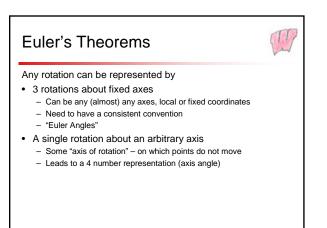
TI,

- Projection onto a subset
- Where do the axes go

 Clearly redundant (if know 1 axis in 2d, figure out the other)



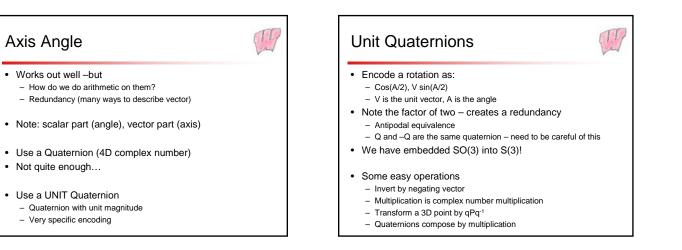
	3x3 Matrices	
pecify easily	NO! (ask an artist to type 9 numbers?)	No
Compact	No! (9 numbers, redundancy)	No
Ensure rotation	Hard (Graham-Schmidth Orthogonalization)	No
Compose	Easy (matrix multiply)	Yes
Inverse	Expensive (Matrix inversion)	Sortof
Transform	Easy and Fast	Yes
nterpolate	No! (really hard)	No
Blend / Average	No!	No
Linear Ops	No!	No
No Singularities	Yes, but metrics are hard to find	Sorto



Euler Angles

- Different conventions are used
 - XYZ graphics
 - ZYX animation (human figures)
 - XZX physics
 Roll, Pitch, Yaw (e.g. local) flying
- Very compact
- R³ -> rotations (so can give sliders to artists)
- Perilous
 - Meanings of later transforms depend on earlier ones (not so easy)
 - Singularities (some nearby transforms may be far away)
 - Can't actually do arithmetic on them

	Euler Angles	3x3
		matrice
Specify easily	Sortof (false sense of security)	No
Compact	Yes (3 numbers)	No
Ensure rotation	Yes (any numbers are a rotation)	No
Compose	No	Yes
Inverse	Sortof (a trivial inverse is one of many)	Sortof
Transform	No (need to form matrices)	Yes
Interpolate	No (interpolating numbers gives weird things)	No
Blend / Average	No (false sense of security)	No
Linear Ops	No (false sense of security)	No



Г

TI,

Interpolation

- · Goal: "nice" paths between orientations
- · Great circle routes
 - Points follow geodesics on spheres (circles)
 - "Smoothest" and "shortest" possible paths
- · Constant velocity (magnitude / speed) along route
- SLERP spherical linear interpolation
- Easy if have a single axis
 Interpolate the angle linearly
- So put into local coordinates of the first, and interpolate
- Kindof expensive

• SLERP is great, but expensive

- Notice: if you scale a vector, it's the same direction

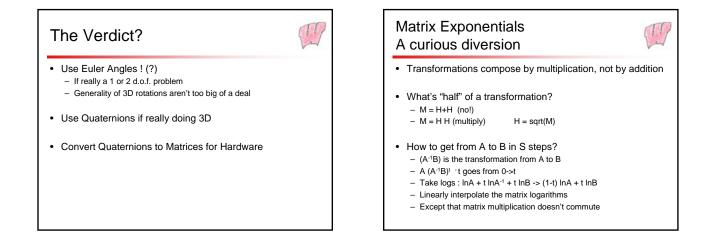
 aV (linearly interpolate a) the "axis of rotation" is still V
- · Linearly interpolate the quaternions
- Renormalize (to make unit quaternions)
- Traces the same great circle route

 But not at the same velocity
- But is VERY cheap and easy to compute

More than 2 Quaternions

- SLERP does not associate
 (A->B)->C is not A->(B->C)
- How to average/blend N Quaternions?
- Mathematically right answers are hard
 Need to understand logarithms
- Normalized LERP works "well enough"
 - Not constant velocity (but this is a small effect)
 - Does associate
- Use exponential coordinates otherwise

	Quaternions	Euler Angles	3x3 matrices
Specify easily	No (but use Euler UI)	Sortof	No
Compact	Yes (4 numbers)	Yes	No
Ensure rotation	Yes (renormalize is easy)	Yes	No
Compose	Yes (very fast quaternion multiply)	No	Yes
Inverse	Yes (very fast)	Sortof	Sortof
Transform	Yes (very fast, about the same at mmult)	No	Yes
Interpolate	Yes (SLERP or NLERP)	No	No
Blend / Average	Yes (NLERP or spherical averages)	No	No
Linear Ops	Yes (NLERP or log maps)	No	No



TI,